# Propositional Logic

## Combinatorial Problem Solving (CPS)

Albert Oliveras  Enric Rodríguez-Carbonell

May 11, 2026

# Overview of the session

- Definition of Propositional Logic
- General Concepts in Logic
  - Reduction to SAT
- CNFs and DNFs
  - Tseitin Transformation
- Problem Solving with SAT
- Resolution

# Definition of Propositional Logic

**SYNTAX** (what is a formula?)**:**

- There is a set $\mathcal{P}$ of propositional variables, usually denoted by (subscripted) $p, q, r, \ldots$
- The set of propositional formulas over $\mathcal{P}$ is defined as:
  - Every propositional variable is a formula
  - If $F$ is a formula, $\neg F$ is also a formula
  - If $F$ and $G$ are formulas, $(F \wedge G)$ is also a formula
  - If $F$ and $G$ are formulas, $(F \vee G)$ is also a formula
  - Nothing else is a formula
- Formulas are usually denoted by (subscripted) $F, G, H, \ldots$
- Examples:

$$p \qquad \neg p \qquad (p \vee q) \qquad \neg(p \wedge q)$$

$$(p \wedge (\neg p \vee q)) \qquad ((p \wedge q) \vee (r \vee \neg q)) \qquad \ldots$$

# Definition of Propositional Logic

**SEMANTICS** (what is an interpretation $I$, when $I$ satisfies $F$?)**:**

- An interpretation $I$ over $\mathcal{P}$ is a function $I : \mathcal{P} \to \{0, 1\}$.
- $eval_I : Formulas \to \{0, 1\}$ is a function defined as follows:
  - $eval_I(p) = I(p)$
  - $eval_I(\neg F) = 1 - eval_I(F)$
  - $eval_I(\ \ (F \wedge G)\ \ ) = \min\{eval_I(F), eval_I(G)\}$
  - $eval_I(\ \ (F \vee G)\ \ ) = \max\{eval_I(F), eval_I(G)\}$
- $I$ satisfies $F$ (written $I \models F$) if and only if $eval_I(F) = 1$.
- If $I \models F$ we say that
  - $I$ is a model of $F$ or, equivalently
  - $F$ is true in $I$.

# Definition of Propositional Logic

**EXAMPLE:**

- Let $F$ be the formula $(p \wedge (q \vee \neg r))$.
- Let $I$ be such that $I(p) = I(r) = 1$ and $I(q) = 0$.
- Let us compute $eval_I(F)$ (use your intuition first!)

- Is there any $I$ such that $I \models F$?

# Definition of Propositional Logic

**EXAMPLE:**

- Let $F$ be the formula $(p \wedge (q \vee \neg r))$.
- Let $I$ be such that $I(p) = I(r) = 1$ and $I(q) = 0$.
- Let us compute $eval_I(F)$ (use your intuition first!)

$$eval_I((p \wedge (q \vee \neg r))) \quad =$$

- Is there any $I$ such that $I \models F$?

# Definition of Propositional Logic

**EXAMPLE:**

- Let $F$ be the formula $(p \wedge (q \vee \neg r))$.
- Let $I$ be such that $I(p) = I(r) = 1$ and $I(q) = 0$.
- Let us compute $eval_I(F)$ (use your intuition first!)

$$eval_I((p \wedge (q \vee \neg r))) \quad = \quad \min\{\, eval_I(p),\ eval_I((q \vee \neg r)) \,\}$$

- Is there any $I$ such that $I \models F$?

# Definition of Propositional Logic

**EXAMPLE:**

- Let $F$ be the formula $(p \wedge (q \vee \neg r))$.
- Let $I$ be such that $I(p) = I(r) = 1$ and $I(q) = 0$.
- Let us compute $eval_I(F)$ (use your intuition first!)

$$\begin{aligned} eval_I((p \wedge (q \vee \neg r))) &= \min\{\, eval_I(p),\ eval_I((q \vee \neg r))\ \} \\ &= \min\{\, eval_I(p),\ \max\{\, eval_I(q), eval_I(\neg r)\}\ \} \end{aligned}$$

- Is there any $I$ such that $I \models F$?

# Definition of Propositional Logic

**EXAMPLE:**

- Let $F$ be the formula $(p \wedge (q \vee \neg r))$.
- Let $I$ be such that $I(p) = I(r) = 1$ and $I(q) = 0$.
- Let us compute $eval_I(F)$ (use your intuition first!)

$$\begin{aligned} eval_I((p \wedge (q \vee \neg r))) &= \min\{\, eval_I(p),\ eval_I((q \vee \neg r))\ \} \\ &= \min\{\, eval_I(p),\ \max\{\, eval_I(q), eval_I(\neg r)\}\ \} \\ &= \min\{\, eval_I(p),\ \max\{\, eval_I(q), 1 - eval_I(r)\}\ \} \end{aligned}$$

- Is there any $I$ such that $I \models F$?

# Definition of Propositional Logic

**EXAMPLE:**

- Let $F$ be the formula $(p \wedge (q \vee \neg r))$.
- Let $I$ be such that $I(p) = I(r) = 1$ and $I(q) = 0$.
- Let us compute $eval_I(F)$ (use your intuition first!)

$$
\begin{aligned}
eval_I((p \wedge (q \vee \neg r))) &= \min\{\, eval_I(p),\ eval_I((q \vee \neg r))\ \} \\
&= \min\{\, eval_I(p),\ \max\{\, eval_I(q), eval_I(\neg r)\}\ \} \\
&= \min\{\, eval_I(p),\ \max\{\, eval_I(q), 1 - eval_I(r)\}\ \} \\
&= \min\{\, I(p),\ \max\{\, I(q), 1 - I(r)\}\ \}
\end{aligned}
$$

- Is there any $I$ such that $I \models F$?

# Definition of Propositional Logic

**EXAMPLE:**

- Let $F$ be the formula $(p \wedge (q \vee \neg r))$.
- Let $I$ be such that $I(p) = I(r) = 1$ and $I(q) = 0$.
- Let us compute $eval_I(F)$ (use your intuition first!)

$$
\begin{aligned}
eval_I((p \wedge (q \vee \neg r))) &= \min\{\, eval_I(p),\ eval_I((q \vee \neg r))\ \} \\
&= \min\{\, eval_I(p),\ \max\{\, eval_I(q), eval_I(\neg r)\}\ \} \\
&= \min\{\, eval_I(p),\ \max\{\, eval_I(q), 1 - eval_I(r)\}\ \} \\
&= \min\{\, I(p),\ \max\{\, I(q), 1 - I(r)\}\ \} \\
&= \min\{\, 1,\ \max\{\, 0, 1 - 1\}\ \}
\end{aligned}
$$

- Is there any $I$ such that $I \models F$?

# Definition of Propositional Logic

**EXAMPLE:**

- Let $F$ be the formula $(p \wedge (q \vee \neg r))$.
- Let $I$ be such that $I(p) = I(r) = 1$ and $I(q) = 0$.
- Let us compute $eval_I(F)$ (use your intuition first!)

$$
\begin{aligned}
eval_I((p \wedge (q \vee \neg r))) &= \min\{\, eval_I(p),\ eval_I((q \vee \neg r))\ \} \\
&= \min\{\, eval_I(p),\ \max\{\, eval_I(q), eval_I(\neg r)\}\ \} \\
&= \min\{\, eval_I(p),\ \max\{\, eval_I(q), 1 - eval_I(r)\}\ \} \\
&= \min\{\, I(p),\ \max\{\, I(q), 1 - I(r)\}\ \} \\
&= \min\{\, 1,\ \max\{\, 0, 1 - 1\}\ \} \\
&= 0
\end{aligned}
$$

- Is there any $I$ such that $I \models F$?

# Definition of Propositional Logic

**EXAMPLE:**

- Let $F$ be the formula $(p \wedge (q \vee \neg r))$.
- Let $I$ be such that $I(p) = I(r) = 1$ and $I(q) = 0$.
- Let us compute $eval_I(F)$ (use your intuition first!)

$$
\begin{aligned}
eval_I((p \wedge (q \vee \neg r))) &= \min\{\, eval_I(p),\ eval_I((q \vee \neg r))\ \} \\
&= \min\{\, eval_I(p),\ \max\{\, eval_I(q), eval_I(\neg r)\}\ \} \\
&= \min\{\, eval_I(p),\ \max\{\, eval_I(q), 1 - eval_I(r)\}\ \} \\
&= \min\{\, I(p),\ \max\{\, I(q), 1 - I(r)\}\ \} \\
&= \min\{\, 1,\ \max\{\, 0, 1 - 1\}\ \} \\
&= 0
\end{aligned}
$$

- Is there any $I$ such that $I \models F$?

# Definition of Propositional Logic

**EXAMPLE:**

- Let $F$ be the formula $(p \wedge (q \vee \neg r))$.
- Let $I$ be such that $I(p) = I(r) = 1$ and $I(q) = 0$.
- Let us compute $eval_I(F)$ (use your intuition first!)

$$
\begin{aligned}
eval_I((p \wedge (q \vee \neg r))) &= \min\{\, eval_I(p),\ eval_I((q \vee \neg r))\ \} \\
&= \min\{\, eval_I(p),\ \max\{\, eval_I(q), eval_I(\neg r)\}\ \} \\
&= \min\{\, eval_I(p),\ \max\{\, eval_I(q), 1 - eval_I(r)\}\ \} \\
&= \min\{\, I(p),\ \max\{\, I(q), 1 - I(r)\}\ \} \\
&= \min\{\, 1,\ \max\{\, 0, 1 - 1\}\ \} \\
&= 0
\end{aligned}
$$

- Is there any $I$ such that $I \models F$?

  YES, $I(p) = I(q) = I(r) = 1$ is a possible model.

# Definition of Propositional Logic

## EXAMPLE

- We have 3 pigeons and 2 holes.
  If each hole can have at most one pigeon,
  is it possible to place all pigeons in the holes?
- Vocabulary: $p_{i,j}$ means $i$-th pigeon is in $j$-th hole
- Each pigeon is placed in at least one hole:

$$(p_{1,1} \vee p_{1,2}) \ \wedge \ (p_{2,1} \vee p_{2,2}) \ \wedge \ (p_{3,1} \vee p_{3,2})$$

- Each hole can hold at most one pigeon:

$$\neg(p_{1,1} \wedge p_{2,1}) \ \wedge \ \neg(p_{1,1} \wedge p_{3,1}) \ \wedge \ \neg(p_{2,1} \wedge p_{3,1}) \ \wedge$$
$$\neg(p_{1,2} \wedge p_{2,2}) \ \wedge \ \neg(p_{1,2} \wedge p_{3,2}) \ \wedge \ \neg(p_{2,2} \wedge p_{3,2})$$

- Resulting formula has no model

# Definition of Propositional Logic

A small syntax extension:

- We will write $(F \to G)$ as an abbreviation for $(\neg F \vee G)$
- Similarly, $(F \leftrightarrow G)$ is an abbreviation of $((F \to G) \wedge (G \to F))$

# Overview of the session

- Definition of Propositional Logic
- General Concepts in Logic
  - Reduction to SAT
- CNFs and DNFs
  - Tseitin Transformation
- Problem Solving with SAT
- Resolution

# General Concepts in Logic

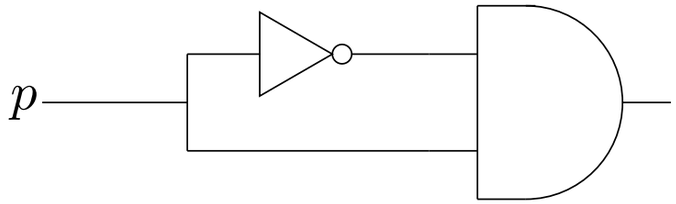
Let $F$ and $G$ be arbitrary formulas. Then:

- $F$ is satisfiable if it has at least one model
- $F$ is unsatisfiable (also a contradiction) if it has no model
- $F$ is a tautology if every interpretation is a model of $F$
- $G$ is a logical consequence of $F$, denoted $F \models G$,
  if every model of $F$ is a model of $G$
- $F$ and $G$ are logically equivalent, denoted $F \equiv G$,
  if $F$ and $G$ have the same models

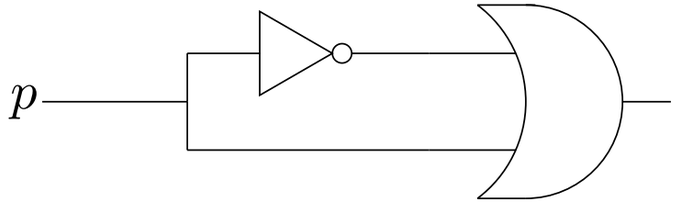
Note that:

- All definitions are only based on the concept of model.
- Hence they are independent of the logic.

# General Concepts in Logic

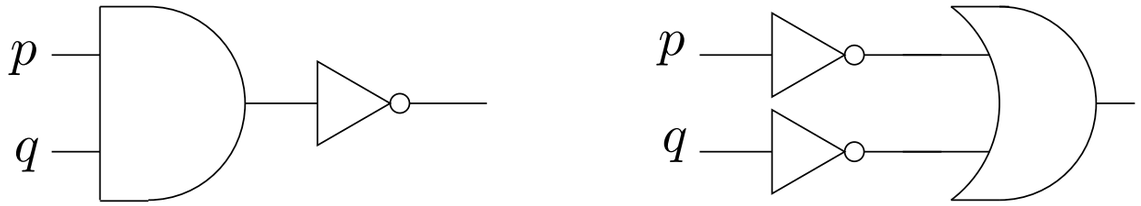



- Circuit corresponds to formula $(\neg p \wedge p)$
- Formula unsatisfiable amounts to *"circuit output is always 0"*



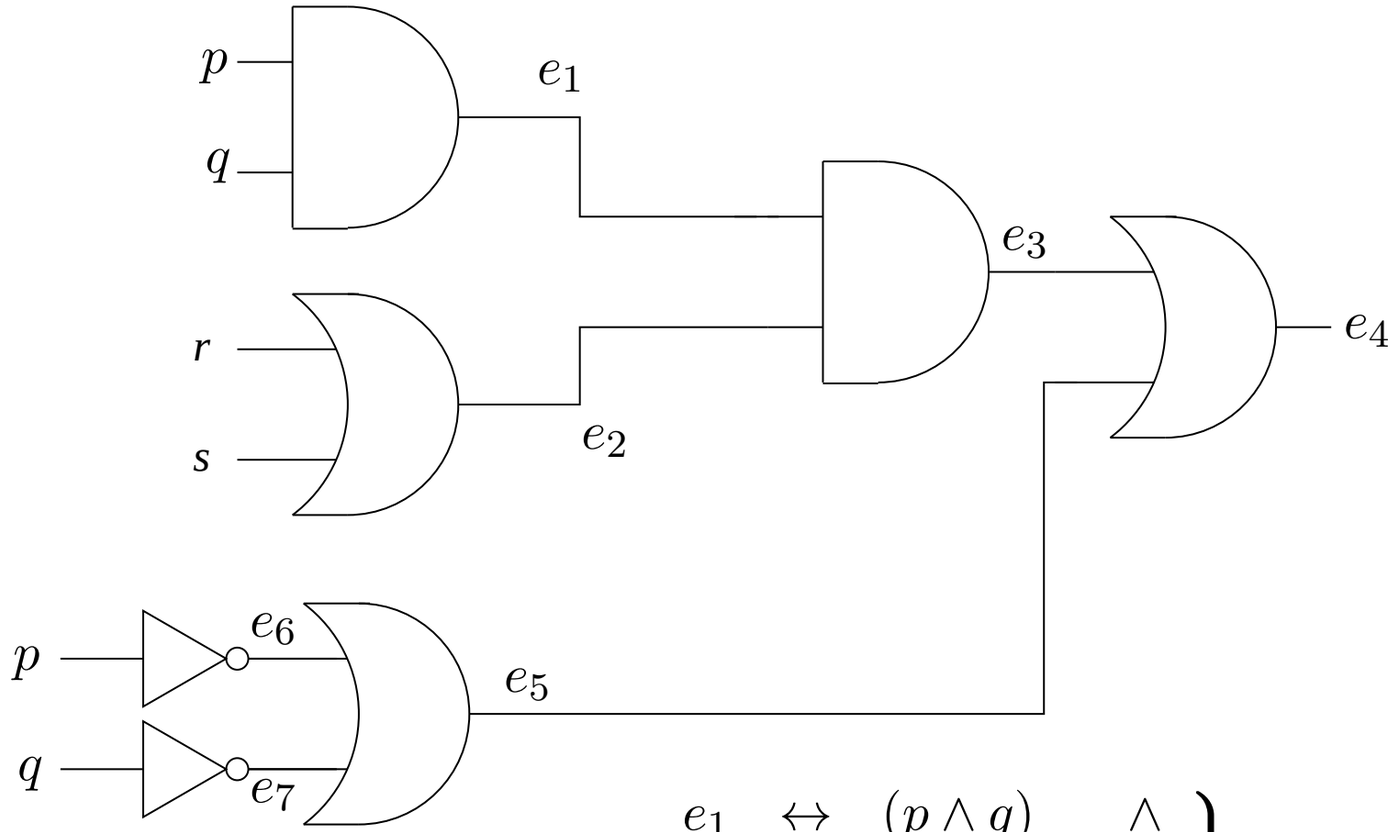

- Circuit corresponds to formula $(\neg p \vee p)$
- Formula is a tautology amounts to *"circuit output is always 1"*

# General Concepts in Logic




- Circuit on the left corresponds to formula $F := \neg(p \wedge q)$
- Circuit on the right corresponds to formula $G := (\neg p \vee \neg q)$
- They are functionally equivalent, i.e. same inputs produce same output
- That corresponds to saying $F \equiv G$
- Cheapest / fastest / least power-consuming circuit is then chosen

# General Concepts in Logic




Is $e_1$ always different from $e_5$?

$e_1 \neq e_5$ in the circuit amounts to

$$\left.\begin{array}{lcll} e_1 & \leftrightarrow & (p \wedge q) & \wedge \\ e_2 & \leftrightarrow & (r \vee s) & \wedge \\ e_3 & \leftrightarrow & (e_1 \wedge e_2) & \wedge \\ e_4 & \leftrightarrow & (e_3 \vee e_5) & \wedge \\ e_5 & \leftrightarrow & (e_6 \wedge e_7) & \wedge \\ e_6 & \leftrightarrow & (\neg p) & \wedge \\ e_7 & \leftrightarrow & (\neg q) & \end{array}\right\} \models e_1 \leftrightarrow \neg e_5$$

# Reduction to SAT

Assume we have a black box **SAT** that given a formula $F$:

- **SAT**$(F) =$ YES iff $F$ is satisfiable
- **SAT**$(F) =$ NO iff $F$ is unsatisfiable

How to reuse **SAT** for detecting tautology, logical consequences, ...?

- $F$ tautology iff **SAT**$(\neg F) =$ NO
- $F \models G$ iff **SAT**$(F \wedge \neg G) =$ NO
- $F \equiv G$ iff **SAT**$((F \wedge \neg G) \vee (\neg F \wedge G)) =$ NO

# Reduction to SAT

Assume we have a black box **SAT** that given a formula $F$:

- **SAT**$(F) =$ YES iff $F$ is satisfiable
- **SAT**$(F) =$ NO iff $F$ is unsatisfiable

How to reuse **SAT** for detecting tautology, logical consequences, ...?

- $F$ **not** taut. iff **SAT**$(\neg F) =$ YES
- $F \not\models G$ iff **SAT**$(F \wedge \neg G) =$ YES
- $F \not\equiv G$ iff **SAT**$((F \wedge \neg G) \vee (\neg F \wedge G)) =$ YES

# Reduction to SAT

Assume we have a black box **SAT** that given a formula $F$:

- **SAT**$(F) =$ YES iff $F$ is satisfiable
- **SAT**$(F) =$ NO iff $F$ is unsatisfiable

How to reuse **SAT** for detecting tautology, logical consequences, ...?

- $F$ tautology iff **SAT**$(\neg F) =$ NO
- $F \models G$ iff **SAT**$(F \wedge \neg G) =$ NO
- $F \equiv G$ iff **SAT**$((F \wedge \neg G) \vee (\neg F \wedge G)) =$ NO

Hence, a single tool suffices: all problems can be reduced to SAT (propositional SATisfiability)

The black box **SAT** will be called a SAT solver

GOAL: learn how to build a SAT solver

# Overview of the session

- Definition of Propositional Logic
- General Concepts in Logic
  - Reduction to SAT
- CNFs and DNFs
  - Tseitin Transformation
- Problem Solving with SAT
- Resolution

# CNFs and DNFs

In order to construct our SAT solver
it will simplify our job to assume that the formula $F$ has a given format.

- A literal is a propositional variable ($p$) or a negation of one ($\neg p$)
- A clause is a disjunction of zero or more literals ($l_1 \vee \ldots l_n$)
- The empty clause (zero literals) is denoted with $\square$ and is unsatisfiable
- A formula is in Conjunctive Normal Form (CNF) if
  it is a conjunction of zero or more disjunctions of literals (i.e., clauses)
- A formula is in Disjunctive Normal Form (DNF) if
  it is a disjunction of zero or more conjunctions of literals (i.e., cubes)

**Examples:**

$p \wedge (q \vee \neg r) \wedge (q \vee p \vee \neg r)$ is in CNF
$p \vee (q \wedge \neg r) \vee (q \wedge p \wedge \neg r)$ is in DNF

# CNFs and DNFs

- Given a formula $F$ there exist formulas
  - $G$ in CNF with $F \equiv G$ and ($G$ is said to be a CNF of $F$)
  - $H$ in DNF with $F \equiv H$ ($H$ is said to be a DNF of $F$)
- Which is the complexity of deciding whether $F$ is satisfiable...
  - ... if $F$ is an arbitrary formula?
  - ... if $F$ is in CNF?
  - ... if $F$ is in DNF?

# CNFs and DNFs

- Given a formula $F$ there exist formulas
  - $G$ in CNF with $F \equiv G$ and ($G$ is said to be a CNF of $F$)
  - $H$ in DNF with $F \equiv H$ ($H$ is said to be a DNF of $F$)
- Which is the complexity of deciding whether $F$ is satisfiable...
  - ... if $F$ is an arbitrary formula? NP-complete (Cook's Theorem)
  - ... if $F$ is in CNF?
  - ... if $F$ is in DNF?

# CNFs and DNFs

- Given a formula $F$ there exist formulas
  - $G$ in CNF with $F \equiv G$ and ($G$ is said to be a CNF of $F$)
  - $H$ in DNF with $F \equiv H$ ($H$ is said to be a DNF of $F$)
- Which is the complexity of deciding whether $F$ is satisfiable...
  - ... if $F$ is an arbitrary formula? NP-complete (Cook's Theorem)
  - ... if $F$ is in CNF? NP-complete (even if clauses have $\leq 3$ literals!)
  - ... if $F$ is in DNF?

# CNFs and DNFs

- Given a formula $F$ there exist formulas
  - $G$ in CNF with $F \equiv G$ and ($G$ is said to be a CNF of $F$)
  - $H$ in DNF with $F \equiv H$ ($H$ is said to be a DNF of $F$)
- Which is the complexity of deciding whether $F$ is satisfiable...
  - ... if $F$ is an arbitrary formula? NP-complete (Cook's Theorem)
  - ... if $F$ is in CNF? NP-complete (even if clauses have $\leq 3$ literals!)
  - ... if $F$ is in DNF? linear

  Procedure **SAT**($F$)

  *Input:* formula $F$ in DNF
  *Output:* **YES** if there exists $I$ such that $I \models F$, **NO** otherwise

  1. If the DNF is empty then return **NO**.
  Else take a cube $C$ of the DNF

  2. If there is a variable $p$ such that both $p$, $\neg p$ appear in $C$,
  then $C$ cannot be made true: remove it and go to step 1.
  Else define $I$ to make $C$ true and return **YES**.

# CNFs and DNFs

- Idea: given $F$, find a DNF of $F$ and apply the linear-time algorithm
- Why this does not work?

# CNFs and DNFs

- Idea: given $F$, find a DNF of $F$ and apply the linear-time algorithm
- Why this does not work? Finding a DNF of $F$ may take exponential time
- In fact there are formulas for which CNFs/DNFs have exponential size

# CNFs and DNFs

- Idea: given $F$, find a DNF of $F$ and apply the linear-time algorithm
- Why this does not work? Finding a DNF of $F$ may take exponential time
- In fact there are formulas for which CNFs/DNFs have exponential size

- Consider $\mathrm{xor}$ defined as $\mathrm{xor}(x_1) = x_1$ and if $n > 1$:

$$\begin{aligned}\mathrm{xor}(x_1, ..., x_n) \quad = \quad & (\mathrm{xor}(x_1, ..., x_{\lfloor \frac{n}{2} \rfloor}) \quad \wedge \quad \neg\mathrm{xor}(x_{\lfloor \frac{n}{2} \rfloor + 1}, ..., x_n)) \vee \\ & (\neg\mathrm{xor}(x_1, ..., x_{\lfloor \frac{n}{2} \rfloor}) \quad \wedge \quad \mathrm{xor}(x_{\lfloor \frac{n}{2} \rfloor + 1}, ..., x_n))\end{aligned}$$

- The size of $\mathrm{xor}(x_1, ..., x_n)$ is $\Theta(n^2)$
- Cubes (conjunctions of literals) of a DNF of $\mathrm{xor}(x_1, ..., x_n)$ have $n$ literals
- Any DNF of $\mathrm{xor}(x_1, ..., x_n)$ has at least $2^{n-1}$ cubes
  (one for each of the assignments with an odd number of 1s)
- Any CNF of $\neg\mathrm{xor}(x_1, ..., x_n)$ also has an exponential number of clauses

# CNFs and DNFs

- Idea: given $F$, find a DNF of $F$ and apply the linear-time algorithm
- Why this does not work? Finding a DNF of $F$ may take exponential time
- In fact there are formulas for which CNFs/DNFs have exponential size

- Consider $\mathrm{xor}$ defined as $\mathrm{xor}(x_1) = x_1$ and if $n > 1$:

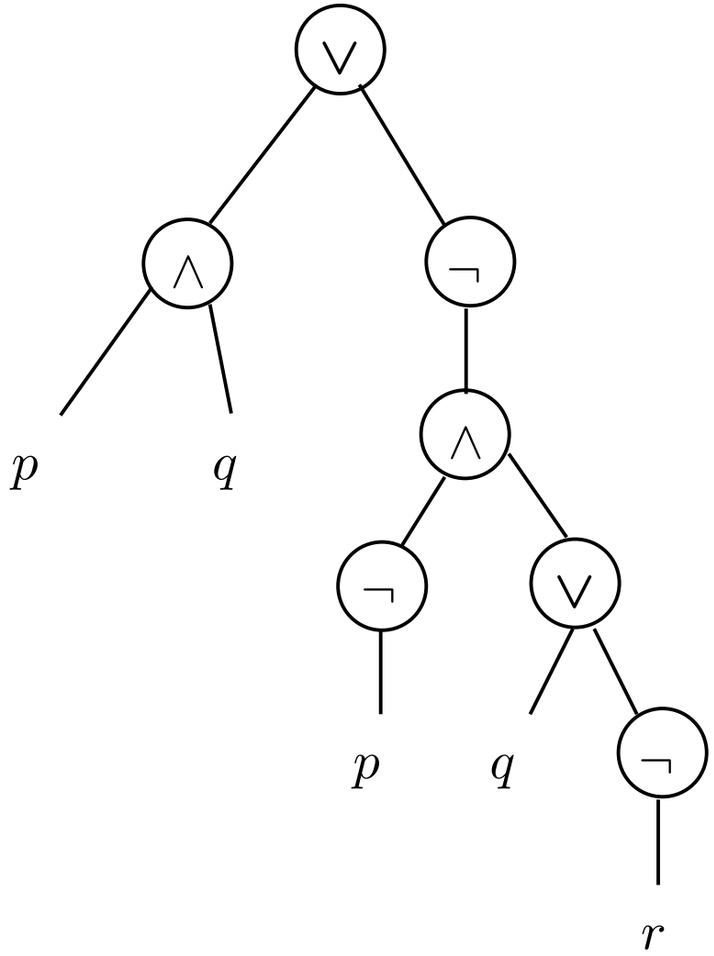
$$\begin{aligned}\mathrm{xor}(x_1, ..., x_n) \quad = \quad & (\mathrm{xor}(x_1, ..., x_{\lfloor \frac{n}{2} \rfloor}) \quad \wedge \quad \neg\mathrm{xor}(x_{\lfloor \frac{n}{2} \rfloor + 1}, ..., x_n)) \vee \\ & (\neg\mathrm{xor}(x_1, ..., x_{\lfloor \frac{n}{2} \rfloor}) \quad \wedge \quad \mathrm{xor}(x_{\lfloor \frac{n}{2} \rfloor + 1}, ..., x_n))\end{aligned}$$

- The size of $\mathrm{xor}(x_1, ..., x_n)$ is $\Theta(n^2)$
- Cubes (conjunctions of literals) of a DNF of $\mathrm{xor}(x_1, ..., x_n)$ have $n$ literals
- Any DNF of $\mathrm{xor}(x_1, ..., x_n)$ has at least $2^{n-1}$ cubes
  (one for each of the assignments with an odd number of 1s)
- Any CNF of $\neg\mathrm{xor}(x_1, ..., x_n)$ also has an exponential number of clauses

- Next we'll see a workaround

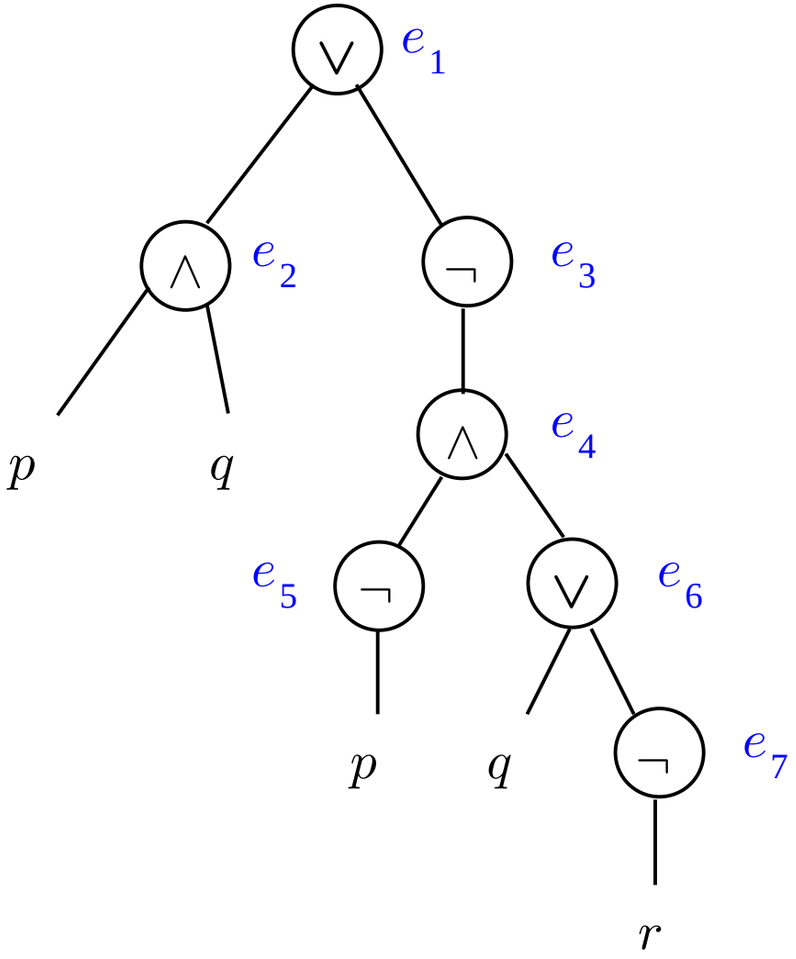
# Tseitin Transformation

Let $F$ be $(p \wedge q) \vee \neg(\ \neg p \wedge (q \vee \neg r)\ )$



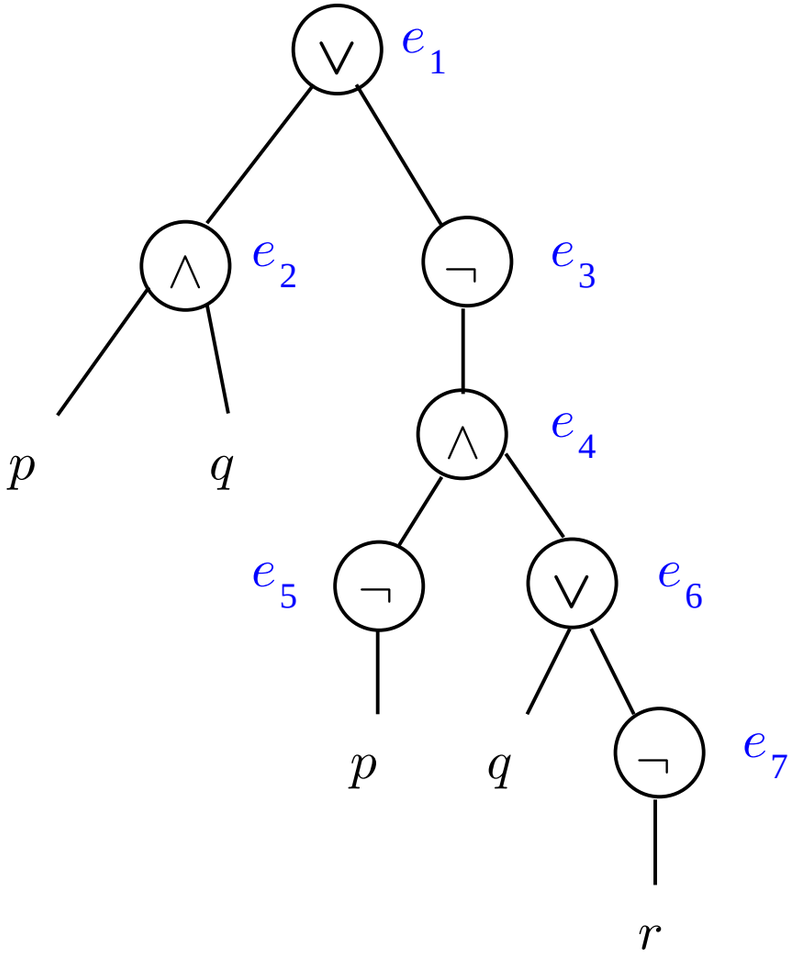
# Tseitin Transformation

Let $F$ be $(p \wedge q) \vee \neg(\ \neg p \wedge (q \vee \neg r)\ )$



# Tseitin Transformation

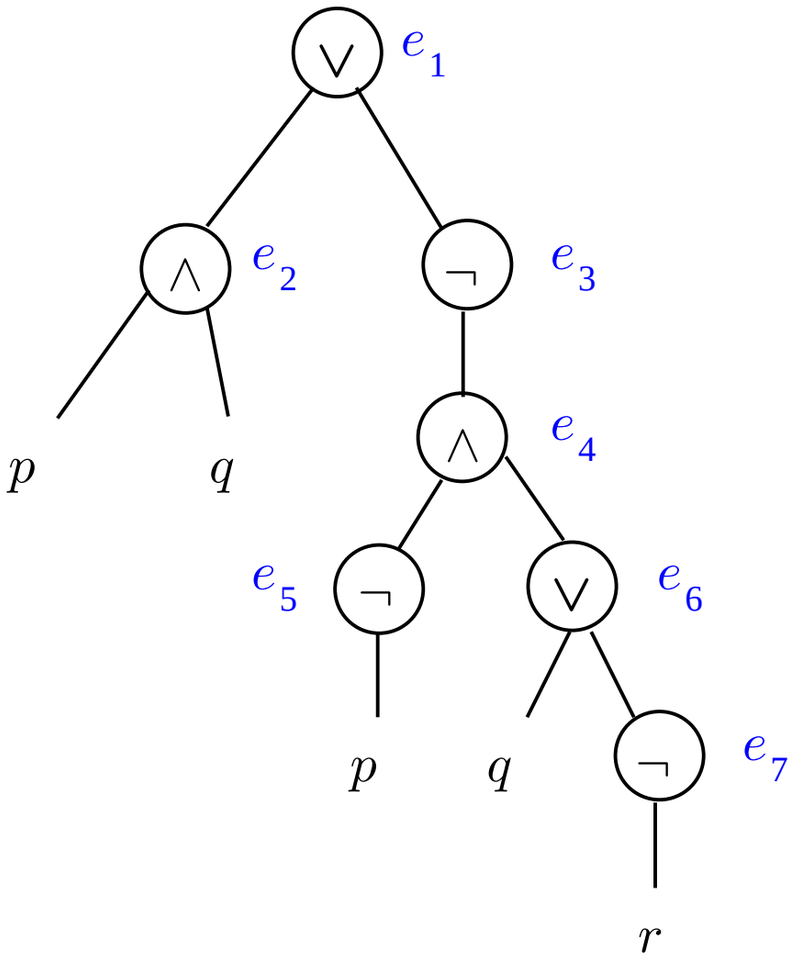
Let $F$ be $(p \wedge q) \vee \neg(\ \neg p \wedge (q \vee \neg r)\ )$




- $e_1$
- $e_1 \leftrightarrow e_2 \vee e_3$
- $e_2 \leftrightarrow p \wedge q$
- $e_3 \leftrightarrow \neg e_4$
- $e_4 \leftrightarrow e_5 \wedge e_6$
- $e_5 \leftrightarrow \neg p$
- $e_6 \leftrightarrow q \vee \neg e_7$
- $e_7 \leftrightarrow \neg r$

# Tseitin Transformation

Let $F$ be $(p \wedge q) \vee \neg(\ \neg p \wedge (q \vee \neg r)\ )$

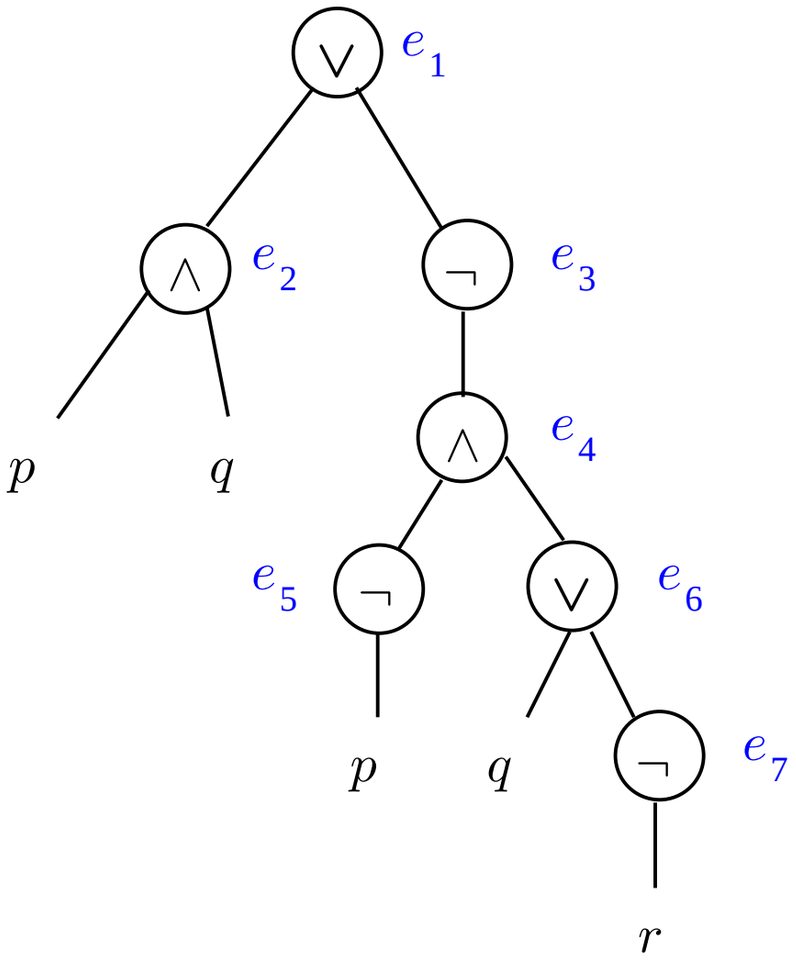



- $e_1$
- $e_1 \leftrightarrow e_2 \vee e_3$

  $\neg e_1 \vee e_2 \vee e_3$
  $\neg e_2 \vee e_1$
  $\neg e_3 \vee e_1$
- $e_2 \leftrightarrow p \wedge q$
- $e_3 \leftrightarrow \neg e_4$
- $e_4 \leftrightarrow e_5 \wedge e_6$
- $e_5 \leftrightarrow \neg p$
- $e_6 \leftrightarrow q \vee \neg e_7$
- $e_7 \leftrightarrow \neg r$

# Tseitin Transformation

Let $F$ be $(p \wedge q) \vee \neg(\ \neg p \wedge (q \vee \neg r)\ )$



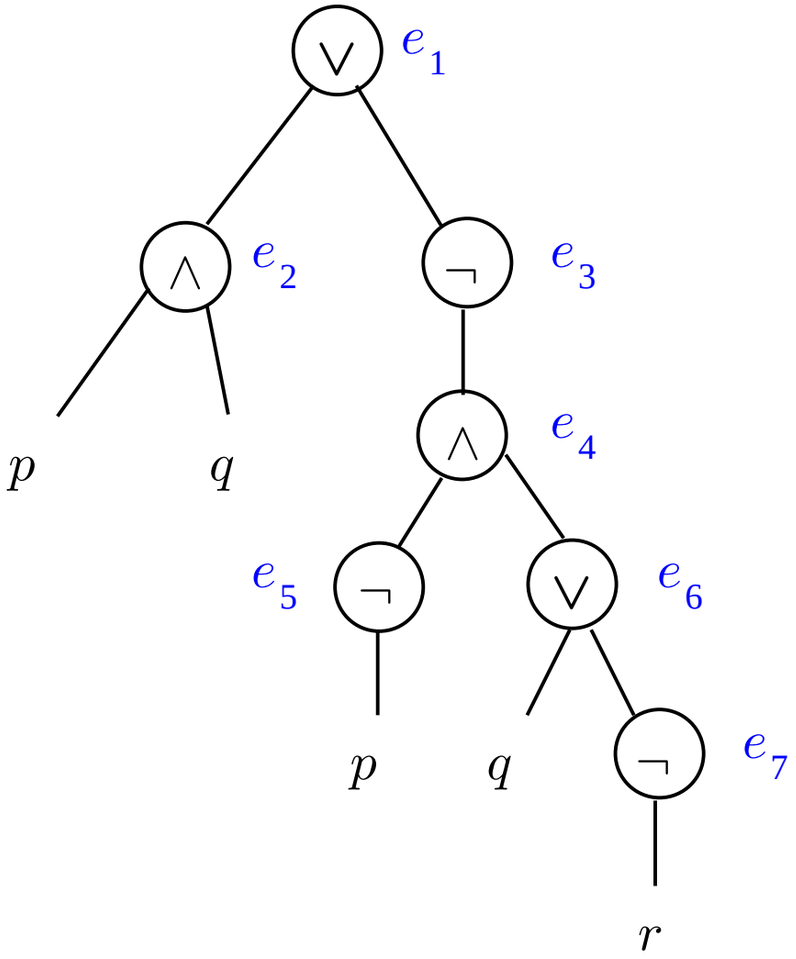

- $e_1$
- $e_1 \leftrightarrow e_2 \vee e_3$

  $\neg e_1 \vee e_2 \vee e_3$
  $\neg e_2 \vee e_1$
  $\neg e_3 \vee e_1$
- $e_2 \leftrightarrow p \wedge q$

  $\neg p \vee \neg q \vee e_2$
  $\neg e_2 \vee p$
  $\neg e_2 \vee q$
- $e_3 \leftrightarrow \neg e_4$
- $e_4 \leftrightarrow e_5 \wedge e_6$
- $e_5 \leftrightarrow \neg p$
- $e_6 \leftrightarrow q \vee \neg e_7$
- $e_7 \leftrightarrow \neg r$

# Tseitin Transformation

Let $F$ be $(p \wedge q) \vee \neg(\ \neg p \wedge (q \vee \neg r)\ )$




- $e_1$
- $e_1 \leftrightarrow e_2 \vee e_3$

  $\neg e_1 \vee e_2 \vee e_3$
  $\neg e_2 \vee e_1$
  $\neg e_3 \vee e_1$
- $e_2 \leftrightarrow p \wedge q$

  $\neg p \vee \neg q \vee e_2$
  $\neg e_2 \vee p$
  $\neg e_2 \vee q$
- $e_3 \leftrightarrow \neg e_4$

  $\neg e_3 \vee \neg e_4$
  $e_3 \vee e_4$
- $e_4 \leftrightarrow e_5 \wedge e_6$
- $e_5 \leftrightarrow \neg p$
- $e_6 \leftrightarrow q \vee \neg e_7$
- $e_7 \leftrightarrow \neg r$

# Tseitin Transformation

- Variations of Tseitin transformation are used in practice in SAT solvers
- Tseitin transformation does not produce an equivalent CNF: for example, the Tseitin transformation of $F = \neg p$ is $G = e \wedge (\neg e \vee \neg p) \wedge (e \vee p)$, and

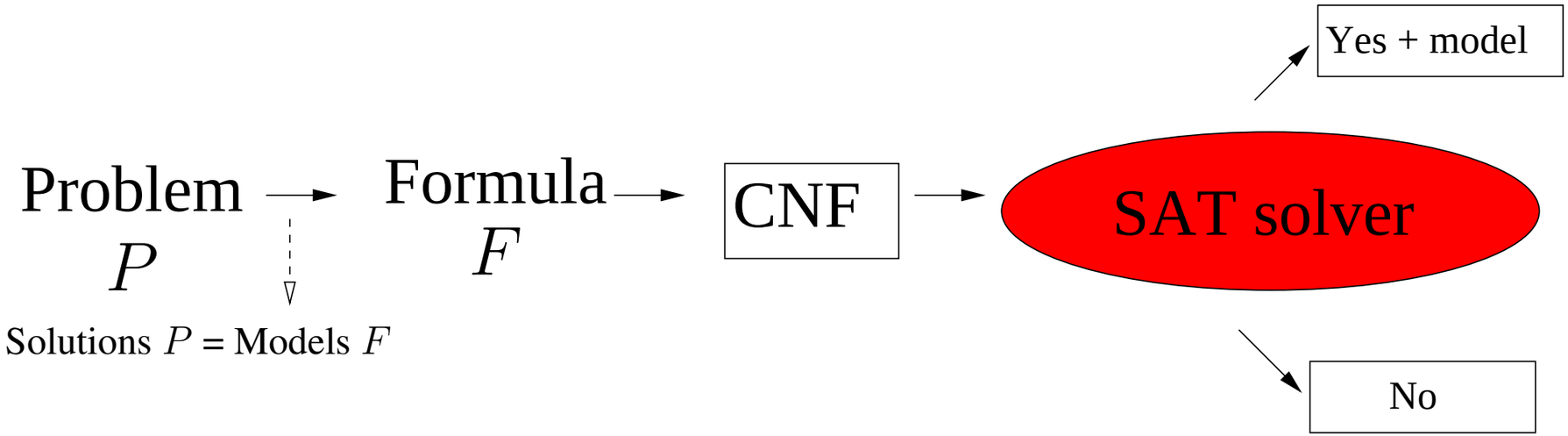
| $e$ | $p$ | $F$ | $G$ |
|---|---|---|---|
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 |

- Still, CNF obtained from $F$ via Tseitin transformation has nice properties:
  - It is equisatisfiable to $F$
  - Any model of CNF projected to the variables in $F$ gives a model of $F$
  - Any model of $F$ can be completed to a model of the CNF
  - Can be computed in linear time in the size of $F$
- Hence no model is lost nor added in the transformation

# Overview of the session

- Definition of Propositional Logic
- General Concepts in Logic
  - Reduction to SAT
- CNFs and DNFs
  - Tseitin Transformation
- Problem Solving with SAT
- Resolution

# Problem Solving with SAT




- This is the standard flow when solving problems with SAT
- Transformation from $P$ to $F$ is called the encoding into SAT

  Already seen some examples: pigeon-hole problem
  Other examples will be seen in the next classes
- CNF transformation already explained
- Let us see now how to design efficient SAT solvers

# Overview of the session

- Definition of Propositional Logic
- General Concepts in Logic
  - Reduction to SAT
- CNFs and DNFs
  - Tseitin Transformation
- Problem Solving with SAT
- Resolution

# Resolution

- The resolution rule is

$$\frac{p \vee C \quad \neg p \vee D}{C \vee D}$$

- $Res(S)$ = closure of set of clauses $S$ under resolution =
  = clauses inferred in zero or more steps of resolution from $S$
- Properties:
  - Resolution is correct:
    $Res(S)$ only contains logical consequences
  - Resolution is refutationally complete:
    if $S$ is unsatisfiable, then $\square \in Res(S)$
  - $Res(S)$ is a finite set of clauses
- So, given a set of clauses $S$, its satisfiability can be checked by:
  1. Computing $Res(S)$
  2. **If** $\square \in Res(S)$ **Then** UNSAT ; **Else** SAT